

DELPHES, a framework for fast simulation of a generic collider experiment

S. Ovyn*, X. Rouby¹

Center for Particle Physics and Phenomenology (CP3), Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Abstract

It is always delicate to know whether theoretical predictions are visible and measurable in a high energy collider experiment due to the complexity of the related detectors, data acquisition chain and software. We introduce here a new C++-based framework, DELPHES, for fast simulation of a general-purpose experiment. The simulation includes a tracking system, embedded into a magnetic field, calorimetry and a muon system, and possible very forward detectors arranged along the beamline. The framework is interfaced to standard file formats (e.g. Les Houches Event File or HepMC) and outputs observable objects for analysis, like missing transverse energy and collections of electrons or jets. The simulation of detector response takes into account the detector resolution, and usual reconstruction algorithms, such as FASTJET. A simplified preselection can also be applied on processed data for trigger emulation. Detection of very forward scattered particles relies on the transport in beamlines with the HECTOR software. Finally, the FROG 2D/3D event display is used for visualisation of the collision final states. An overview of DELPHES is given as well as a few LHC use-cases for illustration.

Preprint: CP3-09-01, arXiv:0903.2225 [hep-ph]



PROGRAM SUMMARY

Program Title: DELPHES

Current version: 1.7

Journal Reference:

Catalogue identifier:

Distribution format: tar.gz

Programming language: C++

External routines/libraries: ROOT environment

Subprograms used: HepMC, STDHEP, FastJet, Hector, FROG. All provided within DELPHES distribution.

URL: <http://www.fynu.ucl.ac.be/delphes.html>

Key words: DELPHES, fast simulation, trigger, event display, LHC, FASTJET, HECTOR, FROG, Les Houches Event File, HepMC, ROOT

PACS: 29.85.-c, 07.05.Tp, 29.90.+r, 29.50.+v

1. Introduction

Experiments at high energy colliders are very complex systems for several reasons. Firstly, in terms of the various detector sub-

systems, including tracking, central calorimetry, forward calorimetry, and muon chambers. Such apparatus differ in their detection principles, technologies, geometrical acceptances, resolutions and sensitivities. Secondly, due to the requirement of a highly effective online selection (i.e. a *trigger*), subdivided into several levels for an optimal reduction factor of “uninteresting” events, but based only on partially processed data. Finally, in terms of the experiment software, with different data formats (like *raw* or *reconstructed*

*Corresponding author: +32.10.47.32.29.

Email addresses: severine.ovyn@uclouvain.be (S. Ovyn),
xavier.rouby@cern.ch (X. Rouby)

¹Now in Physikalisches Institut, Albert-Ludwigs-Universität Freiburg

data), many reconstruction algorithms and particle identification approaches.

This complexity is handled by large collaborations of thousands of people, but the data and the expertise are only available to their members. Real data analyses require a full detector simulation, including transport of the primary and secondary particles through the detector material accounting for the various detector inefficiencies, the dead material, the imperfections and the geometrical details. Moreover, control of the detector calibration and alignment are crucial. Such simulation is very complicated, technical and requires a large CPU power. On the other hand, phenomenological studies, looking for the observability of given signals, may require only fast but realistic estimates of the expected signals and associated backgrounds.

A new framework, called DELPHES [1], is introduced here, for the fast simulation of a general-purpose collider experiment. Using the framework, observables can be estimated for specific signal and background channels, as well as their production and measurement rates. Starting from the output of event generators, the simulation of the detector response takes into account the subdetector resolutions, by smearing the kinematic properties of the final-state particles². Tracks of charged particles and deposits of energy in calorimetric cells (or *calotowers*) are then created.

DELPHES includes the most crucial experimental features, such as (Fig. 1):

1. the geometry of both central and forward detectors,
2. magnetic field for tracks
3. reconstruction of photons, leptons, jets, *b*-jets, τ -jets and missing transverse energy,
4. lepton isolation,
5. trigger emulation,
6. an event display.

Although this kind of approach yields much realistic results than a simple “parton-level” analysis, a fast simulation comes with some limitations. Detector geometry is idealised, being uniform, symmetric around the beam axis, and having no cracks nor dead material. Secondary interactions, multiple scatterings, photon conversion and bremsstrahlung are also neglected.

Four datafile formats can be used as input in DELPHES³. In order to process events from many different generators, the standard Monte Carlo event structures StdHEP [2] and HepMC [3] can be used as an input. Besides, DELPHES can also provide detector response for events read in “Les Houches Event Format” (LHEF [4]) and root files obtained from .hBOOK using the h2root utility from the ROOT framework [5].

DELPHES uses the ExRootAnalysis utility [6] to create output data in a *.root ntuple. This output contains a copy of the

generator-level data (GEN tree), the analysis data objects after reconstruction (Analysis tree), and possibly the results of the trigger emulation (Trigger tree). In option⁴, DELPHES can produce a reduced output file in *.lhco text format, which is limited to the list of the reconstructed high-level objects in the final states.

The program is driven by input cards. The detector card (data/DetectorCard.dat) allows a large spectrum of running conditions by modifying basic detector parameters, including calorimeter and tracking coverage and resolution, thresholds or jet algorithm parameters. The trigger card (data/TriggerCard.dat) lists the user algorithms for the simplified online preselection.

2. Detector simulation

The overall layout of the general-purpose detector simulated by DELPHES is shown in Fig. 2. A central tracking system (TRACKER) is surrounded by an electromagnetic and a hadron calorimeters (ECAL and HCAL, resp.). Two forward calorimeters (FCAL) ensure a larger geometric coverage for the measurement of the missing transverse energy. Finally, a muon system (MUON) encloses the central detector volume. The fast simulation of the detector response takes into account geometrical acceptance of sub-detectors and their finite resolution, as defined in the detector data card⁵. If no such file is provided, predefined values based on “typical” CMS acceptances and resolutions are used⁶. The geometrical coverage of the various subsystems used in the default configuration are summarised in Tab. 1.

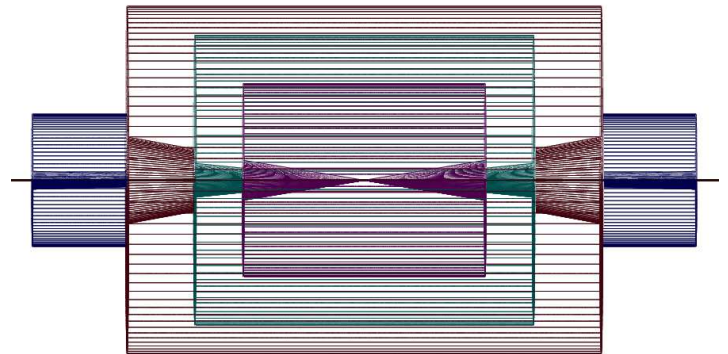


Figure 2: Profile of layout of the generic detector geometry assumed in DELPHES. The innermost layer, close to the interaction point, is a central tracking system (pink). It is surrounded by a central calorimeter volume (green) with both electromagnetic and hadronic sections. The outer layer of the central system (red) consists of a muon system. In addition, two end-cap calorimeters (blue) extend the pseudorapidity coverage of the central detector. The detector parameters are defined in the user-configuration card. The extension of the various subdetectors, as defined in Tab. 1, are clearly visible. The detector is assumed to be strictly symmetric around the beam axis (black line). Additional forward detectors are not depicted.

²Throughout the paper, final-state particles refer as particles considered as stable by the event generator.

³[code] See the HEPEVTConverter, HepMCConverter, LHEFConverter and STDHEPConverter classes.

⁴[code] See the FLAG.lhco variable in the detector datacard. This text file format is shortly described in the user manual.

⁵[code] See the RESOLUTION class.

⁶[code] Detector and trigger cards for the ATLAS and CMS experiments are also provided in data/ directory.

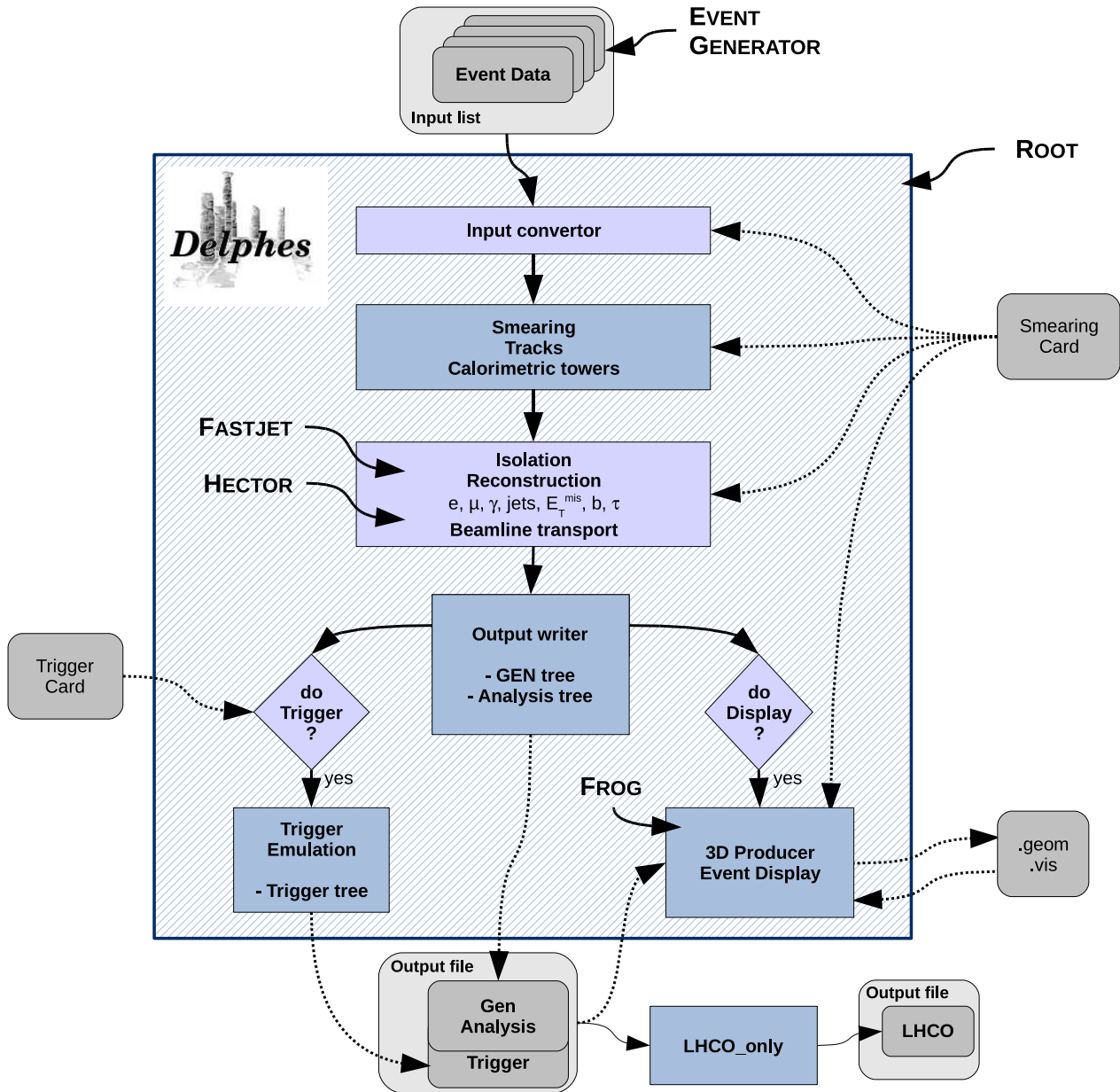


Figure 1: Flow chart describing the principles behind DELPHES. Event files coming from external Monte Carlo generators are read by a converter stage (top). The kinematics variables of the final-state particles are then smeared according to the tunable subdetector resolutions. Tracks are reconstructed in a simulated solenoidal magnetic field and calorimetric towers sample the energy deposits. Based on these low-level objects, dedicated algorithms are applied for particle identification, isolation and reconstruction. The transport of very forward particles to the near-beam detectors is also simulated. Finally, an output file is written, including generator-level and analysis-object data. If requested, a fully parametrisable trigger can be emulated. Optionally, the geometry and visualisation files for the 3D event display can also be produced. All user parameters are set in the *Detector/Smearing Card* and the *Trigger Card*.

Magnetic field

In addition to the subdetectors, the effects of a solenoidal magnetic field are simulated for the charged particles⁷. This affects the position at which charged particles enter the calorimeters and their corresponding tracks.

2.1. Tracks reconstruction

Every stable charged particle with a transverse momentum above some threshold and lying inside the detector volume covered by the tracker provides a track. By default, a track is assumed to be reconstructed with 90% probability⁸ if its transverse momentum p_T is higher than 0.9 GeV/c and if its pseudorapidity $|\eta| \leq 2.5$.

⁷ [code] See the `TrackPropagation` class.

⁸ [code] The reconstruction efficiency is defined in the detector datacard by the `TRACKING_EFF` term.

Table 1: Default extension in pseudorapidity η of the different subdetectors. Full azimuthal (ϕ) acceptance is assumed. The corresponding parameter name, in the detector card, is given.

Subdetector		η	ϕ
TRACKER	CEN_max_tracker	$[-2.5; 2.5]$	$[-\pi; \pi]$
ECAL, HCAL	CEN_max_calor_cen	$[-3.0; 3.0]$	$[-\pi; \pi]$
FCAL	CEN_max_calor_fwd	$[-5; -3] \& [3; 5]$	$[-\pi; \pi]$
MUON	CEN_max_mu	$[-2.4; 2.4]$	$[-\pi; \pi]$

2.2. Simulation of central calorimeters

The energy of each particle considered as stable in the generator particle list is smeared, with a Gaussian distribution depending on the calorimeter resolution. This resolution varies with the sub-calorimeter (ECAL, HCAL, FCAL) measuring the particle. The response of each sub-calorimeter is parametrised as a function of the energy:

$$\frac{\sigma}{E} = \frac{S}{\sqrt{E}} \oplus \frac{N}{E} \oplus C, \quad (1)$$

where S , N and C are the *stochastic*, *noise* and *constant* terms, respectively, and \oplus stands for quadratic additions.

The particle four-momentum p^μ are smeared with a parametrisation directly derived from typical detector technical designs⁹. In the default parametrisation, the calorimeter is assumed to cover the pseudorapidity range $|\eta| < 3$ and consists in an electromagnetic and hadronic parts. Coverage between pseudorapidities of 3.0 and 5.0 is provided by forward calorimeters, with different response to electromagnetic objects (e^\pm, γ) or hadrons. Muons and neutrinos are assumed not to interact with the calorimeters¹⁰. The default values of the stochastic, noise and constant terms are given in Tab. 2.

The energy of electrons and photons found in the particle list are smeared using the ECAL resolution terms. Charged and neutral final-state hadrons interact with the ECAL, HCAL and FCAL. Some long-living particles, such as the K_S^0 and Λ 's, with lifetime $c\tau$ smaller than 10 mm are considered as stable particles although they decay before the calorimeters. The energy smearing of such particles is performed using the expected fraction of the energy, determined according to their decay products, that would be deposited into the ECAL (E_{ECAL}) and into the HCAL (E_{HCAL}). Defining F as the fraction of the energy leading to a HCAL deposit, the two energy values are given by

$$\begin{cases} E_{\text{HCAL}} = E \times F \\ E_{\text{ECAL}} = E \times (1 - F) \end{cases} \quad (2)$$

where $0 \leq F \leq 1$. The electromagnetic part is handled the same way for the electrons and photons. The resulting calorimetry

⁹[code] [7, 8]. The response of the detector is applied to the electromagnetic and the hadronic particles through the `Smearelectron` and `Smearelectron` functions.

¹⁰In the current DELPHES version, particles other than electrons (e^\pm), photons (γ), muons (μ^\pm) and neutrinos (ν_e, ν_μ and ν_τ) are simulated as hadrons for their interactions with the calorimeters. The simulation of stable particles beyond the Standard Model should therefore be handled with care.

Table 2: Default values for the resolution of the central and forward calorimeters. Resolution is parametrised by the *stochastic* (S), *noise* (N) and *constant* (C) terms (Eq. 1). The corresponding parameter name, in the detector card, is given.

Resolution Term	Card flag	Value
ECAL		
S ($\text{GeV}^{1/2}$)	ELG_Scen	0.05
N (GeV)	ELG_Ncen	0.25
C	ELG_Ccen	0.0055
FCAL, electromagnetic part		
S ($\text{GeV}^{1/2}$)	ELG_Sfwd	2.084
N (GeV)	ELG_Nfwd	0
C	ELG_Cfwd	0.107
HCAL		
S ($\text{GeV}^{1/2}$)	HAD_Shcal	1.5
N (GeV)	HAD_Nhcal	0
C	HAD_Chcal	0.05
FCAL, hadronic part		
S ($\text{GeV}^{1/2}$)	HAD_Shf	2.7
N (GeV)	HAD_Nhf	0
C	HAD_Chf	0.13

energy measurement given after the application of the smearing is then $E = E_{\text{HCAL}} + E_{\text{ECAL}}$. For K_S^0 and Λ hadrons¹¹, the energy fraction is F is assumed to be 0.7.

2.3. Calorimetric towers

The smallest unit for geometrical sampling of the calorimeters is a *tower*; it segments the (η, ϕ) plane for the energy measurement. No longitudinal segmentation is available in the simulated calorimeters. All undecayed particles, except muons and neutrinos deposit energy in a calorimetric tower, either in ECAL, in HCAL or FCAL. As the detector is assumed to be cylindrical (e.g. symmetric in ϕ and with respect to the $\eta = 0$ plane), the detector card stores the number of calorimetric towers with $\phi = 0$ and $\eta > 0$ (default: 40 towers). For a given η , the size of the ϕ segmentation is also specified. Fig. 3 illustrates the default segmentation of the (η, ϕ) plane.

¹¹[code] To implement different ratios for other particles, see the `BlockClasses` class.

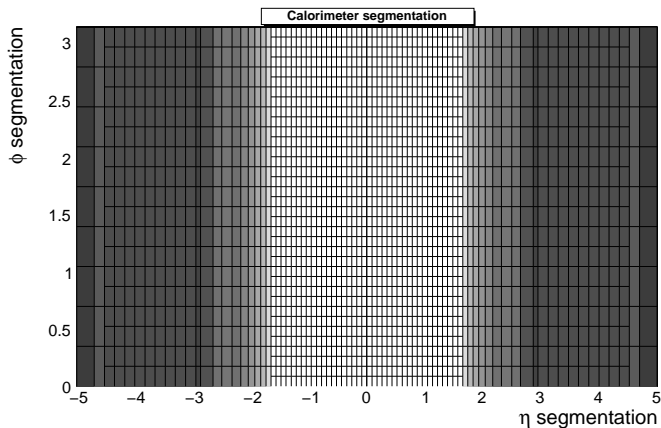


Figure 3: Default segmentation of the calorimeters in the (η, ϕ) plane. Only the central detectors (ECAL, HCAL) and FCAL are considered. ϕ angles are expressed in radians.

The calorimetric towers directly enter in the calculation of the missing transverse energy (MET), and as input for the jet reconstruction algorithms. No sharing between neighbouring towers is implemented when particles enter a tower very close to its geometrical edge. Smearing is applied directly on the accumulated electromagnetic and hadronic energies of each calorimetric tower.

2.4. Very forward detectors simulation

Most of the recent experiments in beam colliders have additional instrumentation along the beamline. These extend the η coverage to higher values, for the detection of very forward final-state particles. Zero Degree Calorimeters (ZDC) are located at zero angle, i.e. are aligned with the beamline axis at the interaction point, and placed beyond the point where the paths of incoming and outgoing beams separate (Fig. 4). These allow the measurement of stable neutral particles (γ and n) coming from the interaction point, with large pseudorapidities (e.g. $|\eta_{n,\gamma}| > 8.3$ in ATLAS and CMS). Forward taggers (called here RP220, for “roman pots at 220 m” and FP420 “for forward proton taggers at 420 m”, as at the LHC) are meant for the measurement of particles following very closely the beam path. To be able to reach these detectors, such particles must have a charge identical to the beam particles, and a momentum very close to the nominal value for the beam. These taggers are near-beam detectors located a few millimetres from the true beam trajectory and this distance defines their acceptance (Tab. 3).

While neutral particles propagate along a straight line to the ZDC, a dedicated simulation of the transport of charged particles is needed for RP220 and FP420. This fast simulation uses the HECTOR software [9], which includes the chromaticity effects and the geometrical aperture of the beamline elements of any arbitrary collider.

Some subdetectors have the ability to measure the time of flight of the particle. This corresponds to the delay after which the particle is observed in the detector, with respect to the bunch crossing reference time at the interaction point (t_0). The time of flight measurement of ZDC and FP420 detector is implemented here. For the

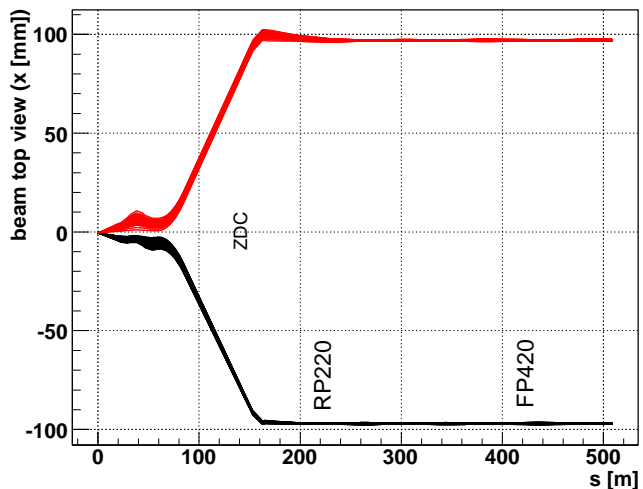


Figure 4: Default location of the very forward detectors, including ZDC and FP420 in the LHC beamline. Incoming (red) and outgoing (black) beams on one side of the interaction point ($s = 0$ m). The Zero Degree Calorimeter is located in perfect alignment with the beamline axis at the interaction point, at 140 m, the beam paths are separated. The forward taggers are near-beam detectors located at 220 m and 420 m. Beamline simulation with HECTOR [9].

ZDC, the formula is simply

$$t = t_0 + \frac{1}{v} \times \left(\frac{s - z}{\cos \theta} \right), \quad (3)$$

where t is the time of flight, t_0 is the true time coordinate of the vertex from which the particle originates, v the particle velocity, s is the ZDC distance to the interaction point, z is the longitudinal coordinate of the vertex from which the particle comes from, θ is the particle emission angle. This assumes that the neutral particle observed in the ZDC is highly relativistic, i.e. travelling at the speed of light c . We also assume that $\cos \theta = 1$, i.e. $\theta \approx 0$ or equivalently η is large. As an example, $\eta = 5$ leads to $\theta = 0.013$ and $1 - \cos \theta < 10^{-4}$. The formula then reduces to

$$t = \frac{1}{c} \times (s - z) \quad (4)$$

For example, a photon takes $0.47 \mu\text{s}$ to reach a ZDC located at $s = 140$ m, neglecting z and θ , and assuming that $v = c$. Only neutrons and photons are currently assumed to be able to reach the ZDC. All other particles are neglected in the ZDC. The ZDCs are composed of an electromagnetic and a hadronic sections, for the measurement of photons and neutrons, respectively. The energy of the observed neutral is smeared according to Eq. 1 and the section resolutions (Tab. 4). The ZDC hits do not enter in the calorimeter tower list used for reconstruction of jets and missing transverse energy.

3. High-level object reconstruction

Analysis object data contain the final collections of particles (e^\pm , μ^\pm , γ) or objects (light jets, b -jets, τ -jets, E_T^{miss}) and are stored¹²

¹²[code] All these processed data are located under the Analysis tree.

Table 3: Default parameters for the forward detectors: distance from the interaction point and detector acceptance. The LHC beamline is assumed around the fifth LHC interaction point (IP). For the zdc, the acceptance depends only on the pseudorapidity η of the particle, which should be neutral and stable. The tagger acceptance is fully determined by the distance in the transverse plane of the detector to the real beam position [9]. It is expressed in terms of the particle energy (E).

Detector	Distance from IP	Acceptance
ZDC	140 m	$ \eta > 8.3$ for n and γ
RP220	220 m	$E \in [6100; 6880]$ (GeV) at 2 mm
FP420	420 m	$E \in [6880; 6980]$ (GeV) at 4 mm

Table 4: Default values for the resolution of the zero degree calorimeters. Resolution is parametrised by the *stochastic* (S), *noise* (N) and *constant* (C) terms (Eq. 1). The corresponding parameter name, in the detector card, is given.

Resolution Term	Card flag	Value
zdc, electromagnetic part		
S (GeV ^{1/2})	ELG_Szdc	0.7
N (GeV)	ELG_Nzdc	0.0
C	ELG_Czdc	0.08
zdc, hadronic part		
S (GeV ^{1/2})	HAD_Szdc	1.38
N (GeV)	HAD_Nzdc	0
C	HAD_Czdc	0.13

in the output file created by DELPHES. In addition, some detector data are added: tracks, calorimetric towers and hits in ZDC, RP220 and FP420. While electrons, muons and photons are easily identified, some other objects are more difficult to measure, like jets or missing energy due to invisible particles.

For most of these objects, their four-momentum and related quantities are directly accessible in DELPHES output (E , \vec{p} , p_T , η and ϕ). Additional properties are available for specific objects (like the charge and the isolation status for e^\pm and μ^\pm , the result of application of b -tag for jets and time-of-flight for some detector hits).

3.1. Photon and charged lepton reconstruction

From here onwards, *electrons* refer to both positrons (e^+) and electrons (e^-), and *charged leptons* refer to electrons and muons (μ^\pm), leaving out the τ^\pm leptons as they decay before being detected.

Electrons and photons

Electron (e^\pm) and photon candidates are reconstructed if they fall into the acceptance of the tracking system and have a transverse momentum above a threshold (default $p_T > 10$ GeV/ c). A calorimetric tower will be seen in the detector, as electrons will leave in addition a track. Subsequently, electrons and photons create a candidate in the jet collection. Assuming a good measurement of the track parameters in the real experiment, the electron energy can be reasonably recovered. In DELPHES, electron energy is smeared according to the resolution of the calorimetric tower where it points to, but independently from any other deposited energy is this tower. This approach is still conservative as the calorimeter resolution is worse than the tracker one.

Muons

Generator-level muons entering the detector acceptance are considered as candidates for the analysis level. The acceptance is defined in terms of a transverse momentum threshold to be overpassed that should be computed using the chosen geometry of the detector and the magnetic field considered (default : $p_T > 10$ GeV/ c) and of the pseudorapidity coverage of the muon system (default: $-2.4 \leq \eta \leq 2.4$). The application of the detector resolution on the muon momentum depends on a Gaussian smearing of the p_T variable¹³. Neither η nor ϕ variables are modified beyond the calorimeters: no additional magnetic field is applied. Multiple scattering is neglected. This implies that low energy muons have in DELPHES a better resolution than in a real detector. Furthermore, muons leave no deposit in calorimeters.

Charged lepton isolation

To improve the quality of the contents of the charged lepton collections, additional criteria can be applied such as isolation. This requires that electron or muon candidates are isolated in the detector from any other particle, within a small cone. In DELPHES, charged lepton isolation demands that there is no other charged particle with $p_T > 2$ GeV/ c within a cone of $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} < 0.5$ around the lepton. The result (i.e. *isolated* or *not*) is added to the charged lepton measured properties. In addition, the sum P_T of the transverse momenta of all tracks but the lepton one within the isolation cone is provided¹⁴:

$$P_T = \sum_{i \neq \mu}^{\text{tracks}} p_T(i)$$

No calorimetric isolation is applied, but the muon collection contains also the ratio ρ_μ between (1) the sum of the transverse energies in all calotowers in a $N \times N$ grid around the muon, and (2) the muon transverse momentum¹⁵:

$$\rho_\mu = \frac{\sum_i E_T(i)}{p_T(\mu)}, \quad i \text{ in } N \times N \text{ grid centred on } \mu.$$

¹³ [code] See the SmearMuon method.

¹⁴ [code] See the IsolFlag and IsolPt values in the Electron or Muon collections in the Analysis tree, as well as the ISOL_PT and ISOL_Cone variables in the detector card.

¹⁵ [code] Calorimetric isolation parameters in the detector card are ISOL_Calo_ET and ISOL_Calo_Grid.

Forward neutrals

The zero degree calorimeter hits correspond to neutral particles with a lifetime long enough to reach these detectors (default: $c\tau \geq 140$ m) and very large pseudorapidities (default: $|\eta| > 8.3$). In current versions of DELPHES, only photons and neutrons are considered. Photons are identified thanks to the electromagnetic section of the calorimeter, and if their energy overpasses a given threshold (def. 20 GeV). Similarly, neutrons are reconstructed according to the resolution of the hadronic section, if their energy exceeds a threshold¹⁶ (def. 50 GeV).

3.2. Jet reconstruction

A realistic analysis requires a correct treatment of particles which have hadronised. Therefore, the most widely currently used jet algorithms have been integrated into the DELPHES framework using the FASTJET tools [10]. Six different jet reconstruction schemes are available¹⁷. The first three belong to the cone algorithm class while the last three are using a sequential recombination scheme. For all of them, the towers are used as input for the jet clustering. Jet algorithms differ in their sensitivity to soft particles or collinear splittings, and in their computing speed performances. By default, reconstruction uses a cone algorithm with $\Delta R = 0.7$. Jets are stored if their transverse energy is higher¹⁸ than 20 GeV.

Cone algorithms

1. *CDF Jet Clusters* [11]: Algorithm forming jets by associating together towers lying within a circle (default radius $\Delta R = 0.7$) in the (η, ϕ) space. This so-called JETCLU cone jet algorithm is used by the CDF experiment in Run II. All towers with a transverse energy E_T higher than a given threshold (default: $E_T > 1$ GeV) are used to seed the jet candidates. The existing FASTJET code has been modified to allow easy modification of the tower pattern in (η, ϕ) space. In following versions of DELPHES, a new dedicated plug-in will be created on this purpose¹⁹.
2. *CDF MidPoint* [12]: Algorithm developed for the CDF Run II to reduce infrared and collinear sensitivities compared to purely seed-based cone by adding ‘midpoints’ (energy barycentres) in the list of cone seeds.
3. *Seedless Infrared Safe Cone* [13]: The SISCONE algorithm is simultaneously insensitive to additional soft particles and collinear splittings, and fast enough to be used in experimental analysis.

Recombination algorithms

The three sequential recombination jet algorithms are safe with respect to soft radiations (*infrared*) and collinear splittings. They rely on recombination schemes where calorimeter tower pairs are

successively merged. The definitions of the jet algorithms are similar except for the definition of the *distances* d used during the merging procedure. Two such variables are defined: the distance d_{ij} between each pair of towers (i, j) , and a variable d_{iB} (*beam distance*) depending on the transverse momentum of the tower i .

The jet reconstruction algorithm browses the calotower list. It starts by finding the minimum value d_{\min} of all the distances d_{ij} and d_{iB} . If d_{\min} is a d_{ij} , the towers i and j are merged into a single tower with a four-momentum $p^\mu = p^\mu(i) + p^\mu(j)$ (*E-scheme recombination*). If d_{\min} is a d_{iB} , the tower is declared as a final jet and is removed from the input list. This procedure is repeated until no towers are left in the input list. Further information on these jet algorithms is given here below, using k_{ti} , y_i and ϕ_i as the transverse momentum, rapidity and azimuth of calotower i and $\Delta R_{ij} = \sqrt{(y_i - y_j)^2 + (\phi_i - \phi_j)^2}$ as the jet-radius parameter:

4. *Longitudinally invariant k_t jet* [14]:

$$\begin{aligned} d_{ij} &= \min(k_{ti}^2, k_{tj}^2) \Delta R_{ij}^2 / R^2 \\ d_{iB} &= k_{ti}^2 \end{aligned} \quad (5)$$

5. *Cambridge/Aachen jet* [15]:

$$\begin{aligned} d_{ij} &= \Delta R_{ij}^2 / R^2 \\ d_{iB} &= 1 \end{aligned} \quad (6)$$

6. *Anti k_t jet* [16]: where hard jets are exactly circular in the (y, ϕ) plane

$$\begin{aligned} d_{ij} &= \min(1/k_{ti}^2, 1/k_{tj}^2) \Delta R_{ij}^2 / R^2 \\ d_{iB} &= 1/k_{ti}^2 \end{aligned} \quad (7)$$

Energy flow

In jets, several particle can leave their energy into a given calorimetric tower, which broadens the jet energy resolution. However, the energy of charged particles associated to jets can be deduced from their reconstructed track, thus providing a way to identify some of the components of towers with multiple hits. When the *energy flow* is switched on in DELPHES²⁰, the energy of tracks pointing to calotowers is extracted and smeared separately, before running the chosen jet reconstruction algorithm. This option allows a better jet E reconstruction.

3.3. b -tagging

A jet is tagged as b -jets if its direction lies in the acceptance of the tracker and if it is associated to a parent b -quark. By default, a b -tagging efficiency of 40% is assumed if the jet has a parent b quark. For c -jets and light jets (i.e. originating in u, d, s quarks or in gluons), a fake b -tagging efficiency of 10% and 1% respectively is assumed²¹. The (mis)tagging relies on the true particle identity (PID) of the most energetic particle within a cone around the observed (η, ϕ) region, with a radius equal to the one used to reconstruct the jet (default: ΔR of 0.7). In current version of DELPHES, the displacement of secondary vertices is not simulated.

¹⁶ [code] These thresholds are defined by the ZDC_gamma_E and ZDC_n_E variables in the detector card.

¹⁷ [code] The choice is done by allocating the JET_jetalgo input parameter in the detector card.

¹⁸ [code] PTCUT_jet variable in the detector card.

¹⁹ [code] JET_coneradius and JET_seed variables in the detector card.

²⁰ [code] Set JET_EFlow to 1 or 0 in the detector card in order to switch on or off the energy flow for jet reconstruction.

²¹ [code] Corresponding to the BTAG_b, BTAG_mistag_c and BTAG_mistag_l constants, for (respectively) the efficiency of tagging of a b -jet, the efficiency of mistagging a c -jet as a b -jet, and the efficiency of mistagging a light jet (u, d, s, g) as a b -jet.

3.4. τ identification

Jets originating from τ -decays are identified using a procedure consistent with the one applied in a full detector simulation [7]. The tagging relies on two properties of the τ lepton. First, 77% of the τ hadronic decays contain only one charged hadron associated to a few neutrals (Tab. 5). Tracks are useful for this criterion. Secondly, the particles arisen from the τ lepton produce narrow jets in the calorimeter (this is defined as the jet *collimation*).

Table 5: Branching ratios for τ^- lepton [17]. h^\pm and h^0 refer to charged and neutral hadrons, respectively. $n \geq 0$ and $m \geq 0$ are integers.

Leptonic decays	
$\tau^- \rightarrow e^- \bar{\nu}_e \nu_\tau$	17.9%
$\tau^- \rightarrow \mu^- \bar{\nu}_\mu \nu_\tau$	17.4%
Hadronic decays	
$\tau^- \rightarrow h^- n \times h^\pm m \times h^0 \nu_\tau$	64.7%
$\tau^- \rightarrow h^- m \times h^0 \nu_\tau$	50.1%
$\tau^- \rightarrow h^- h^+ h^- m \times h^0 \nu_\tau$	14.6%

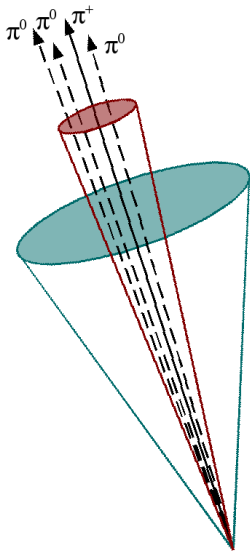


Figure 5: Illustration of the identification of τ -jets (1-prong). The jet cone is narrow and contains only one track. The small cone shown as the red one is used for the *electromagnetic collimation*, while the green cone is the cone radius used to reconstruct the jet originating from the τ -decay.

Electromagnetic collimation

To use the narrowness of the τ -jet, the *electromagnetic collimation* C_τ is defined as the sum of the energy of towers in a small cone of radius R^{em} around the jet axis, divided by the energy of the reconstructed jet. To be taken into account, a calorimeter tower should have a transverse energy E_T^{tower} above a given threshold. A large fraction of the jet energy is expected in this small cone. This fraction, or *collimation factor*, is represented in Fig. 6 for the default values (see Tab. 6).

Table 6: Default values for parameters used in τ -jet reconstruction algorithm. Electromagnetic collimation requirements involve the inner *small* cone radius R^{em} , the minimum transverse energy for calorimeters E_T^{tower} and the collimation factor C_τ . Tracking isolation constrains the number of tracks with a significant transverse momentum p_T^{tracks} in a cone of radius R^{tracks} . Finally, the τ -jet collection is purified by the application of a cut on the p_T of τ -jet candidates.

Parameter	Card flag	Value
Electromagnetic collimation		
R^{em}	TAU_energy_scone	0.15
min E_T^{tower}	JET_M_seed	1.0 GeV
C_τ	TAU_energy_frac	0.95
Tracking isolation		
R^{tracks}	TAU_track_scone	0.4
min p_T^{tracks}	PTAU_track_pt	2 GeV/c
τ -jet candidate		
min p_T	TAUJET_pt	10 GeV/c

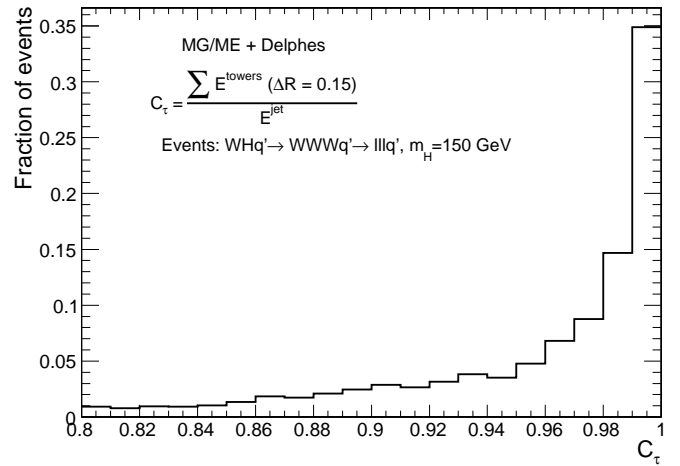


Figure 6: Distribution of the electromagnetic collimation C_τ variable for true τ -jets, normalised to unity. This distribution is shown for associated WH photoproduction [18], where the Higgs boson decays into a W^+W^- pair. Each W boson decays into a $\ell\nu_\ell$ pair, where $\ell = e, \mu, \tau$. Events generated with MADGRAPH/MADEVENT [19]. Final state hadronisation is performed by PYTHIA [20]. Histogram entries correspond to true τ -jets, matched with generator-level data.

Tracking isolation

The tracking isolation for the τ identification requires that the number of tracks associated to particles with significant transverse momenta is one and only one in a cone of radius R^{tracks} (3-prong τ -jets are dropped). This cone should be entirely incorporated into the tracker to be taken into account. Default values of these parameters are given in Tab. 6.

Purity

Once both electromagnetic collimation and tracking isolation are applied, a threshold on the p_T of the τ -jet candidate is requested to purify the collection. This procedure selects τ leptons decaying hadronically with a typical efficiency of 66%.

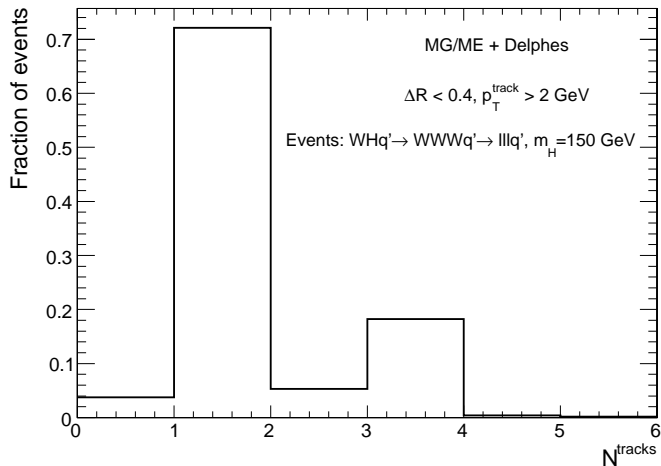


Figure 7: Distribution of the number of tracks N^{tracks} within a small jet cone for true τ -jets, normalised to unity. Photoproduced WH events, where W bosons decay leptonically (e, μ, τ), as in Fig. 6. Histogram entries correspond to true τ -jets, matched with generator-level data.

3.5. Missing transverse energy

In an ideal detector, momentum conservation imposes the transverse momentum of the observed final state \vec{p}_T^{obs} to be equal to the \vec{p}_T vector sum of the invisible particles, written \vec{p}_T^{miss} .

$$\vec{p}_T = \begin{pmatrix} p_x \\ p_y \end{pmatrix} \text{ and } \begin{cases} p_x^{\text{miss}} = -p_x^{\text{obs}} \\ p_y^{\text{miss}} = -p_y^{\text{obs}} \end{cases} \quad (8)$$

The *true* missing transverse energy, i.e. at generator-level, is calculated as the opposite of the vector sum of the transverse momenta of all visible particles – or equivalently, to the vector sum of invisible particle transverse momenta. In a real experiment, calorimeters measure energy and not momentum. Any problem affecting the detector (dead channels, misalignment, noisy towers, cracks) worsens directly the measured missing transverse energy \vec{E}_T^{miss} . In this document, MET is based on the calorimetric towers and only muons and neutrinos are not taken into account for its evaluation²²:

$$\vec{E}_T^{\text{miss}} = - \sum_i^{\text{towers}} \vec{E}_T(i) \quad (9)$$

4. Trigger emulation

New physics in collider experiment are often characterised in phenomenology by low cross-section values, compared to the Standard Model (SM) processes.

As only a tiny fraction of the observed events can be stored for subsequent *offline* analyses, a very large data rejection factor should be applied directly as the events are produced. This data selection

is supposed to reject only well-known SM events²³. Dedicated algorithms of this *online* selection, or *trigger*, should be fast and very efficient for data rejection, in order to preserve the experiment output bandwidth. They must also be as inclusive as possible to avoid losing interesting events.

Most of the usual trigger algorithms select events containing objects (i.e. jets, particles, MET) with an energy scale above some threshold. This is often expressed in terms of a cut on the transverse momentum of one or several objects of the measured event. Logical combinations of several conditions are also possible. For instance, a trigger path could select events containing at least one jet and one electron such as $p_T^{\text{jet}} > 100 \text{ GeV}/c$ and $p_T^e > 50 \text{ GeV}/c$.

A trigger emulation is included in DELPHES, using a fully parametrisable *trigger table*²⁴. When enabled, this trigger is applied on analysis-object data. In a real experiment, the online selection is often divided into several steps (or *levels*). This splits the overall reduction factor into a product of smaller factors, corresponding to the different trigger levels. This is related to the architecture of the experiment data acquisition chain, with limited electronic buffers requiring a quick decision for the first trigger level. First-level triggers are then fast and simple but based only on partial data as not all detector front-ends are readable within the decision latency. Higher level triggers are more complex, of finer-but-not-final quality and based on full detector data.

Real triggers are thus intrinsically based on reconstructed data with a worse resolution than final analysis data. On the contrary, same data are used in DELPHES for trigger emulation and for final analyses.

5. Validation

DELPHES performs a fast simulation of a collider experiment. Its performances in terms of computing time and data size are directly proportional to the number of simulated events and on the considered physics process. As an example, 10,000 $pp \rightarrow t\bar{t}X$ events are processed in 110 s on a regular laptop and use less than 250 MB of disk space. The quality and validity of the output are assessed by comparing to resolution of the reconstructed data to the CMS detector expectations.

Electrons and muons are by construction equal to the experiment designs, as the Gaussian smearing of their kinematics properties is defined according to their resolutions. Similarly, the b -tagging efficiency (for real b -jets) and misidentification rates (for fake b -jets) are taken from the expected values of the experiment. Unlike these simple objects, jets and missing transverse energy should be carefully cross-checked.

5.1. Jet resolution

The majority of interesting processes at the LHC contain jets in the final state. The jet resolution obtained using DELPHES is therefore a crucial point for its validation. Even if DELPHES contains

²²However, as tracks and calorimetric towers are available in the output file, the missing transverse energy can always be reprocessed a posteriori

²³However, some bandwidth is allocated to minimum-bias and/or zero-bias (“random”) triggers that stores a small fraction of the events without any selection criteria.

²⁴[code] The trigger card is the `data/TriggerCard.dat` file.

six algorithms for jet reconstruction, we use here the jet clustering algorithm (JETCLU) with $R = 0.7$ to validate the jet collection.

This validation is based on $pp \rightarrow gg$ events produced with MADGRAPH/MADEVENT and hadronised using PYTHIA [19, 20]. The events were arranged in 14 bins of gluon transverse momentum \hat{p}_T . In each \hat{p}_T bin, every jet in DELPHES is matched to the closest jet of generator-level particles, using the spatial separation between the two jet axes

$$\Delta R = \sqrt{(\eta^{\text{rec}} - \eta^{\text{MC}})^2 + (\phi^{\text{rec}} - \phi^{\text{MC}})^2} < 0.25. \quad (10)$$

The jets made of generator-level particles, here referred as *MC jets*, are obtained by applying the same clustering algorithm to all particles considered as stable after hadronisation. Jets produced by DELPHES and satisfying the matching criterion are called hereafter *reconstructed jets*.

The ratio of the transverse energies of every reconstructed jet E_T^{rec} and its corresponding MC jet E_T^{MC} is calculated in each \hat{p}_T bin. The $E_T^{\text{rec}}/E_T^{\text{MC}}$ histogram is fitted with a Gaussian distribution in the interval ± 2 RMS centred around the mean value. The resolution in each \hat{p}_T bin is obtained by the fit mean $\langle x \rangle$ and variance $\sigma^2(x)$:

$$\sigma\left(\frac{E_T^{\text{rec}}}{E_T^{\text{MC}}}\right)_{\text{fit}}(\hat{p}_T(i)), \text{ for all } i. \quad (11)$$

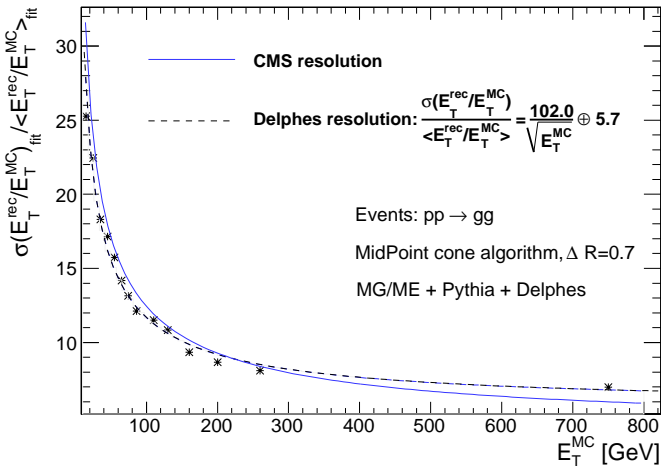


Figure 8: Resolution of the transverse energy of reconstructed jets E_T^{rec} as a function of the transverse energy of the closest jet of generator-level particles E_T^{MC} . The maximum separation between the reconstructed and MC jets is $\Delta R = 0.25$. Pink line is the fit result for comparison to the CMS resolution [7], in blue.

The resulting jet resolution as a function of E_T^{MC} is shown in Fig. 8. This distribution is fitted with a function of the following form:

$$\frac{a}{E_T^{\text{MC}}} \oplus \frac{b}{\sqrt{E_T^{\text{MC}}}} \oplus c, \quad (12)$$

where a , b and c are the fit parameters. It is then compared to the resolution published by the CMS collaboration [7]. The resolution curves from DELPHES and CMS are in good agreement.

5.2. MET resolution

All major detectors at hadron colliders have been designed to be as much hermetic as possible in order to detect the presence of one or more neutrinos and/or new weakly interacting particles through apparent missing transverse energy. The resolution of the \vec{E}_T^{miss} variable, as obtained with DELPHES, is then crucial.

The samples used to study the MET performance are identical to those used for the jet validation. It is worth noting that the contribution to E_T^{miss} from muons is negligible in the studied sample. The input samples are divided in five bins of scalar E_T sums (ΣE_T). This sum, called *total visible transverse energy*, is defined as the scalar sum of transverse energy in all towers. The quality of the MET reconstruction is checked via the resolution on its horizontal component E_x^{miss} .

The E_x^{miss} resolution is evaluated in the following way. The distribution of the difference between E_x^{miss} in DELPHES and at generator-level is fitted with a Gaussian function in each (ΣE_T) bin. The fit RMS gives the MET resolution in each bin. The resulting value is plotted in Fig. 9 as a function of the total visible transverse energy.

The resolution σ_x of the horizontal component of MET is observed to behave like

$$\sigma_x = \alpha \sqrt{E_T} \quad (\text{GeV}^{1/2}), \quad (13)$$

where the α parameter depends on the resolution of the calorimeters.

The MET resolution expected for the CMS detector for similar events is $\sigma_x = (0.6 - 0.7) \sqrt{E_T} \text{ GeV}^{1/2}$ with no pile-up²⁵ [7], which compares very well with the $\alpha = 0.63$ obtained with DELPHES. Similarly, for an ATLAS-like detector, a value of 0.56 is obtained by DELPHES for the α parameter, while the experiment expects it in the range [0.53 ; 0.57] [8].

5.3. τ -jet efficiency

Due to the complexity of their reconstruction algorithm, τ -jets have also to be checked. Table 7 lists the reconstruction efficiencies for the hadronic τ -jets in the CMS experiment and in DELPHES. Agreement is good enough to validate also this reconstruction.

Table 7: Reconstruction efficiencies of τ -jets in decays from Z or H bosons, in DELPHES and in the CMS experiment [21].

CMS		
$Z \rightarrow \tau^+ \tau^-$	38%	
$H \rightarrow \tau^+ \tau^-$	36%	$m_H = 150 \text{ GeV}/c^2$
$H \rightarrow \tau^+ \tau^-$	47%	$m_H = 300 \text{ GeV}/c^2$
DELPHES		
$H \rightarrow \tau^+ \tau^-$	42%	$m_H = 140 \text{ GeV}/c^2$

²⁵Pile-up events are extra simultaneous pp collision occurring at high-luminosity in the same bunch crossing.

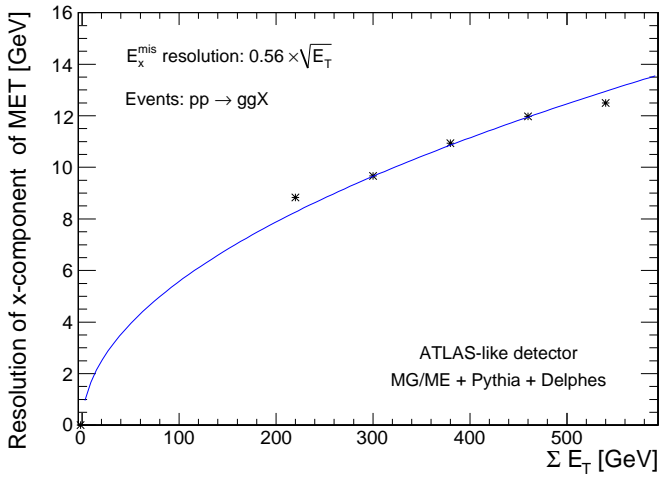
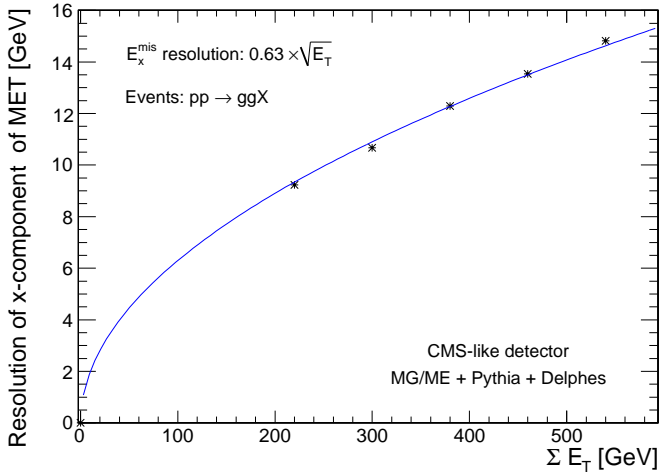


Figure 9: $\sigma(E_x^{\text{mis}})$ as a function on the scalar sum of all towers (ΣE_T) for $pp \rightarrow gg$ events, for a CMS-like detector (top) and an ATLAS-like detector (bottom), for dijet events produced with MadGraph/MadEvent and hadronised with Pythia.

6. Visualisation

When performing an event analysis, a visualisation tool is useful to convey information about the detector layout and the event topology in a simple way. The *Fast and Realistic OpenGL Displayer* FROG [22] has been interfaced in DELPHES, allowing an easy display of the defined detector configuration²⁶.

Two and three-dimensional representations of the detector configuration can be used for communication purposes, as they clearly illustrate the geometric coverage of the different detector subsystems. As an example, the generic detector geometry assumed in this paper is shown in Fig. 2 and 10. The extensions of the central tracking system, the central calorimeters and both forward calorimeters are visible. Note that only the geometrical coverage is depicted and that the calorimeter segmentation is not taken into account in the drawing of the detector. Moreover, both the radius

²⁶ [code] To prepare the visualisation, the FLAG_frog parameter should be equal to 1.

and the length of each sub-detectors are just display parameters and are not relevant for the physics simulation.

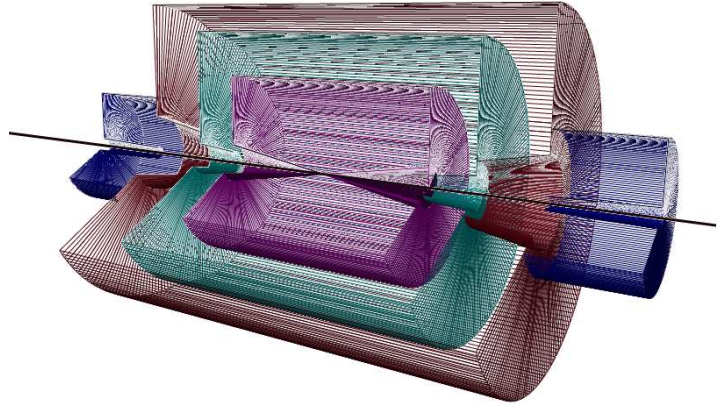


Figure 10: Layout of the generic detector geometry assumed in DELPHES. Open 3D-view of the detector with solid volumes. Same colour codes as for Fig. 2 are applied. Additional forward detectors are not depicted.

Deeper understanding of interesting physics processes is possible by displaying the events themselves. The visibility of each set of objects (e^\pm , μ^\pm , τ^\pm , jets, transverse missing energy) is enhanced by a colour coding. Moreover, kinematics information of each object is visible by a simple mouse action. As an illustration, an associated photoproduction of a W boson and a t quark is shown in Fig. 11. This corresponds to a $pp(\gamma p \rightarrow Wt)pX$ process, where the Wt couple is induced by an incoming photon emitted by one of the colliding proton [23]. This leading proton survives after photon emission and is present in the final state. As the energy and virtuality of the emitted photon are low, the surviving proton does not leave the beam and escapes from the central detector without being detected. The experimental signature is a lack of hadronic activity in the forward hemisphere where the surviving proton escapes. The t quark decays into a W boson and a b quark. Both W bosons decay into leptons ($W \rightarrow \mu\nu_\mu$ and $W \rightarrow e\nu_e$). The balance between the missing transverse energy and the charged lepton pair is clear, as well as the presence of an empty forward region. It is interesting to notice that the reconstruction algorithms build a fake τ -jet around the electron.

For comparison, Fig. 12 depicts an inclusive gluon pair production $pp \rightarrow ggX$. The event final state contains more jets, in particular along the beam axis, which is expected as the interacting protons are destroyed by the collision. Two muon candidates and large missing transverse energy are also visible.

7. Conclusion and perspectives

We have described here the major features of the DELPHES framework, introduced for the fast simulation of a collider experiment. This framework is a tool meant for feasibility studies in phenomenology, gauging the observability of model predictions in collider experiments.

DELPHES takes as an input the output of event-generators and yields analysis-object data in the form of TTree in a root file. The

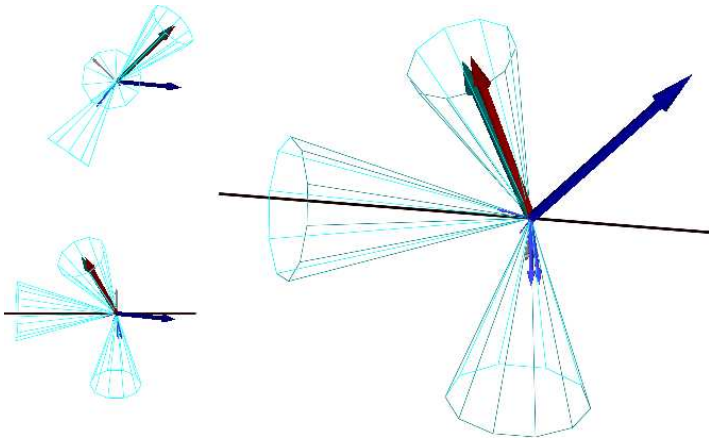


Figure 11: Example of $pp(\gamma p \rightarrow Wt)pY$ event display in different orientations, with $t \rightarrow Wb$. One W boson decays into a $\mu\nu_\mu$ pair and the second one into a $e\nu_e$ pair. The surviving proton leaves a forward hemisphere with no hadronic activity. The isolated muon is shown as the dark blue vector. Around the electron, in red, is reconstructed a fake τ -jet (green vector surrounded by a blue cone), while the reconstructed missing energy (in grey) is very small. One jet is visible in one forward region, along the beamline axis, opposite to the direction of the escaping proton.

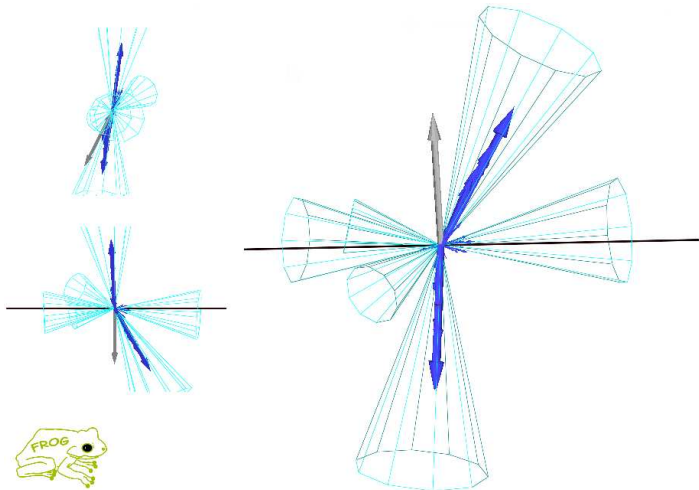


Figure 12: Example of inclusive gluon pair production $pp \rightarrow ggX$. Many jets are visible in the event, in particular along the beam axis. Two muons (in blue) are produced and the missing transverse energy is significant in this event (grey vector).

simulation includes central and forward detectors to produce realistic observables using standard reconstruction algorithms. Moreover, the framework allows trigger emulation and 3D event visualisation.

DELPHES has been developed using the parameters of the CMS experiment but can be easily extended to ATLAS and other non-LHC experiments, as at Tevatron or at the ILC. Further developments include a more flexible design for the subdetector assembly and possibly the implementation of an event mixing module for pile-up event simulation.

This framework has already been used for several analyses, in particular in photon-induced interactions at the LHC [23, 24, 25].

Acknowledgements

The authors would like to thank very warmly Vincent Lemaître for the interesting suggestions during the development of the software, as well as Jérôme de Favereau, Christophe Delaere, Muriel Vander Donckt and David d'Enterria for useful discussions and comments, and Loic Quertenmont for support in interfacing FROG. We are also really grateful to Alice Dechambre and Simon de Visscher for being beta testers of the complete package. Part of this work was supported by the Belgian Federal Office for Scientific, Technical and Cultural Affairs through the Interuniversity Attraction Pole P6/11.

References

- [1] DELPHES, www.fynu.ucl.ac.be/delphes.html
- [2] L.A. Garren, M. Fischler, cepa.fnal.gov/psm/stdhep/c++
- [3] M. Dobbs and J.B. Hansen, **Comput. Phys. Commun.** **134** (2001) 41.
- [4] J. Alwall, et al., **Comput. Phys. Commun.** **176**:300-304,2007.
- [5] R. Brun, F. Rademakers, Nucl. Inst. & Meth. in **Phys. Res. A** **389** (1997) 81-86.
- [6] P. Demin, (2006), unpublished. Now part of MADGRAPH/MADEVENT.
- [7] The CMS Collaboration, **CERN/LHCC 2006-001**.
- [8] The ATLAS Collaboration, **CERN-OPEN 2008-020**, arXiv:0901.0512v1 [hep-ex].
- [9] X. Rouby, J. de Favereau, K. Piotrzkowski, **JINST** **2** P09005 (2007).
- [10] M. Cacciari, G.P. Salam, **Phys. Lett. B** **641** (2006) 57.
- [11] F. Abe et al. (CDF Coll.), **Phys. Rev. D** **45** (1992) 1448.
- [12] G.C. Blazey, et al., arXiv:0005012 [hep-ex].
- [13] G.P. Salam, G. Soyez, **JHEP** **05** (2007) 086.
- [14] S. Catani, Y.L. Dokshitzer, M.H. Seymour, B.R. Webber, **Nucl. Phys. B** **406** (1993) 187; S.D. Ellis, D.E. Soper, **Phys. Rev. D** **48** (1993) 3160.
- [15] Y.L. Dokshitzer, G.D. Leder, S. Moretti, B.R. Webber, **JHEP** **08** (1997) 001; M. Wobisch, T. Wengler, arXiv:9907280 [hep-ph].
- [16] M. Cacciari, G.P. Salam, G. Soyez, **JHEP** **04** (2008) 063.
- [17] C. Amshler et al. (Particle Data Group), **Phys. Lett. B** **667** (2008) 1.
- [18] S. Ovin, **Nucl. Phys. Proc. Suppl.** **179-180** (2008) 269-276.
- [19] J. Alwall, et al., **JHEP** **09** (2007) 028.
- [20] T. Sjostrand, S. Mrenna, P. Skands, **JHEP** **05** (2006) 026.
- [21] R. Kinnunen, A.N. Nikitenko, **CMS NOTE** 1997/002.
- [22] L. Quertenmont, V. Roberfroid, **CMS CR** 2009/028, arXiv:0901.2718v1 [hep-ex].
- [23] J. de Favereau de Jeneret, S. Ovin, **Nucl. Phys. Proc. Suppl.** **179-180** (2008) 277-284; S. Ovin, J. de Favereau de Jeneret, **Nuovo Cimento B**, arXiv:0806.4841 [hep-ph].
- [24] J. de Favereau et al, **CP3-08-04** (2008), to be published in EPJ.
- [25] S. de Visscher, J.M. Gerard, M. Herquet, V. Lemaître, F. Maltoni, arXiv:0904.0705 [hep-ph].
- [26] P. Lebrun, L. Garren, Copyright (c) 1994-1995 Universities Research Association, Inc.

A. User manual

The available C++-code is compressed in a zipped tar file which contains everything needed to run the DELPHES package, assuming a running root installation. The package includes ExRootAnalysis [6], HECTOR [9], FASTJET [10], and FROG [22], as well as the conversion codes to read standard stdHEP input files (mcfio and stdhep) [26] and HEPMC [3]. In order to visualise the events with the FROG software, a few additional external libraries may be required, as explained in <http://projects.hepforge.org/frog/>.

A.1. Getting started

In order to run DELPHES on your system, first download its sources and compile them:

```
wget http://www.fynu.ucl.ac.be/users/s.ovyn/Delphes/files/Delphes_V_*.tar.gz
```

Replace the * symbol by the proper version number²⁷.

```
me@mylaptop:~$ tar -xvf Delphes_V_*.tar.gz
me@mylaptop:~$ cd Delphes_V_*.
me@mylaptop:~$ ./genMakefile.tcl > Makefile
me@mylaptop:~$ make
```

Due to the large number of external utilities, the number of printed lines during the compilation can be high. The user should not pay attention to possible warning messages, which are due to the external packages used by DELPHES. When compilation is completed, the following message is printed:

```
me@mylaptop:~$ Delphes has been compiled
me@mylaptop:~$ Ready to run
```

A.2. Running DELPHES on your events

In this sub-appendix, we will explain how to use DELPHES to perform a fast simulation of a general-purpose detector on your event files. The first step to use DELPHES is to create the list of input event files (e.g. inputlist.list). It is important to notice that all the files comprised in the list file should have the same of extension (*.hep, *.lhe, *.hepmc or *.root). In the simplest way to run DELPHES, you need this input file and you need to specify the name of the output file that will contain the generator-level data (GEN tree), the analysis data objects after reconstruction (Analysis tree), and the results of the trigger emulation (Trigger tree).

```
me@mylaptop:~$ ./Delphes inputlist.list OutputRootFileName.root
```

A.2.1. Setting up the configuration

The program is driven by two datacards (default cards are data/DetectorCard.dat and data/TriggerCard.dat) which allow the user to choose among a large spectrum of running conditions. Please note that if the user does not provide these datacards, the running will be done using the default parameters defined in the constructor of the class RESOLUTION (see next). If you choose a different detector or running configuration, you will need to edit the datacards accordingly. Detector and trigger cards are provided in the data/ subdirectory for the CMS and ATLAS experiments.

1. The detector card

It contains all pieces of information needed to run DELPHES:

- detector parameters, including calorimeter and tracking coverage and resolutions, transverse energy thresholds for object reconstruction and jet algorithm parameters.
- six flags (FLAG_bfield, FLAG_vfd, FLAG_RP, FLAG_trigger, FLAG_frog and FLAG_lhco), should be set in order to configure the magnetic field propagation, the very forward detectors simulation, the use of very forward taggers, the trigger selection, the preparation for Frog display and the creation of an output file in *.lhco text format (respectively).

If no datacard is provided by the user, the default smearing and running parameters are used:

```
# Detector extension, in pseudorapidity units (|eta|)
CEN_max_tracker    2.5    // Maximum tracker coverage
CEN_max_calor_cen  3.0    // central calorimeter coverage
CEN_max_calor_fwd  5.0    // forward calorimeter pseudorapidity coverage
CEN_max_mu         2.4    // muon chambers pseudorapidity coverage
```

²⁷Refer to the download page on the DELPHES website <http://www.fynu.ucl.ac.be/users/s.ovyn/Delphes/download.html>. Current version of DELPHES for this manual is V 1.7 (May 2009)

```

# Energy resolution for electron/photon
#  $\sigma/E = C + N/E + S/\sqrt{E}$ , E in GeV
ELG_Scen      0.05    // S term for central ECAL
ELG_Ncen      0.25    // N term for central ECAL
ELG_Ccen      0.005   // C term for central ECAL
ELG_Sfwd      2.084   // S term for FCAL
ELG_Nfwd      0.      // N term for FCAL
ELG_Cfwd      0.107   // C term for FCAL
ELG_Szdc      0.70    // S term for ZDC
ELG_Nzdc      0.      // N term for ZDC
ELG_Czdc      0.08    // C term for ZDC

# Energy resolution for hadrons in ecal/hcal/hf
#  $\sigma/E = C + N/E + S/\sqrt{E}$ , E in GeV
HAD_Shcal     1.5     // S term for central HCAL
HAD_Nhcal     0.      // N term for central HCAL
HAD_Chcal     0.05    // C term for central HCAL
HAD_Shf       2.7     // S term for FCAL
HAD_Nhf       0.      // N term for FCAL
HAD_Chf       0.13    // C term for FCAL
HAD_Szdc      1.38    // S term for ZDC
HAD_Nzdc      0.      // N term for ZDC
HAD_Czdc      0.13    // C term for ZDC

# Time resolution for ZDC/RP220/RP420
ZDC_T_resolution 0 // in s
RP220_T_resolution 0 // in s
RP420_T_resolution 0 // in s

# Muon smearing
MU_SmearPt      0.01 // transverse momentum Pt in GeV/c

# Tracking efficiencies
TRACK_ptmin     0.9   // minimal pT
TRACK_eff       90    // efficiency associated to the tracking (%)

# Calorimetric towers
TOWER_number    40
### list of the edges of each tower in eta for eta>0 assuming
### a symmetric detector in eta<0
### the list starts with the lower edge of the most central tower
### the list ends with the higher edged of the most forward tower
### there should be NTOWER+1 values
TOWER_eta_edges 0. 0.087 0.174 0.261 0.348 0.435 0.522 0.609 0.696 0.783
                0.870 0.957 1.044 1.131 1.218 1.305 1.392 1.479 1.566 1.653
                1.740 1.830 1.930 2.043 2.172 2.322 2.500 2.650 2.868 2.950
                3.125 3.300 3.475 3.650 3.825 4.000 4.175 4.350 4.525 4.700
                5.000

### list of the tower size in phi (in degrees), assuming that all
### towers are similar in phi for a given eta value
### the list starts with the phi-size of the most central tower (eta=0)
### the list ends with the phi-size of the most forward tower
### there should be NTOWER values
TOWER_dphi 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
           10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 20 20

```

```

# Thresholds for reconstructed objects, in GeV/c
PTCUT_elec      10.0
PTCUT_muon     10.0
PTCUT_jet      20.0
PTCUT_gamma    10.0
PTCUT_taujet   10.0

# Thresholds for reconstructed objects in ZDC, E in GeV
ZDC_gamma_E    20
ZDC_n_E        50

# Charged lepton isolation. Pt and Et in GeV
ISOL_PT        2.0 //minimal pt of tracks for isolation criteria
ISOL_Cone      0.5 //Cone for isolation criteria
ISOL_Calo_ET   2.0 //minimal tower E_T for isolation criteria. 1E99 means "off"
ISOL_Calo_Grid 3   //Grid size (N x N) for calorimetric isolation

# General jet variable
JET_coneradius 0.7 // generic jet radius
JET_jetalgo    1   // 1 for Cone algorithm,
                // 2 for MidPoint algorithm,
                // 3 for SIScone algorithm,
                // 4 for kt algorithm
                // 5 for Cambridge/Aachen algorithm
                // 6 for anti-kt algorithm
JET_seed       1.0 // minimum seed to start jet reconstruction, in GeV
JET_Eflow     1   // Energy flow: perfect energy assumed in the tracker coverage.
                // 1 is 'on' ; 0 is 'off'

# Tagging definition
BTAG_b        40   // b-tag efficiency (%)
BTAG_mistag_c 10   // mistagging (%)
BTAG_mistag_l 1   // mistagging (%)

# FLAGS
FLAG_bfield   1   //1 to run the bfield propagation else 0
FLAG_vfd     1   //1 to run the very forward detectors else 0
FLAG_RP      1   //1 to run the very forward detectors else 0
FLAG_trigger 1   //1 to run the trigger selection else 0
FLAG_frog    1   //1 to run the FROG event display
FLAG_lhco    1   //1 to run the LHCO

# In case BField propagation allowed
TRACK_radius  129   // radius of the BField coverage, in cm
TRACK_length  300   // length of the BField coverage, in cm
TRACK_bfield_x 0    // X component of the BField, in T
TRACK_bfield_y 0    // Y component of the BField, in T
TRACK_bfield_z 3.8  // Z component of the BField, in T

# Very forward detector extension, in pseudorapidity
# if allowed
VFD_min_zdc   8.3   // Zero-Degree neutral Calorimeter
VFD_s_zdc     140   // distance of the ZDC, from the IP, in [m]

#Hector parameters
RP_220_s      220   // distance of the RP to the IP, in meters
RP_220_x      0.002 // distance of the RP to the beam, in meters

```

```

RP_420_s      420      // distance of the RP to the IP, in meters
RP_420_x      0.004    // distance of the RP to the beam, in meters
RP_beam1Card  data/LHCB1IR5_v6.500.tfs // beam optics file, beam 1
RP_beam2Card  data/LHCB2IR5_v6.500.tfs // beam optics file, beam 2
RP_IP_name    IP5      // tag for IP in Hector ; 'IP1' for ATLAS
RP_offsetEl_x 0.097    // horizontal separation between both beam, in meters
RP_offsetEl_y 0        // vertical separation between both beam, in meters
RP_offsetEl_s 120     // distance of beam separation point, from IP
RP_cross_x    -500    // IP offset in horizontal plane, in micrometers
RP_cross_y    0       // IP offset in vertical plane, in micrometers
RP_cross_ang_x 142.5  // half-crossing angle in horizontal plane, in microrad
RP_cross_ang_y 0      // half-crossing angle in vertical plane, in microrad

# In case FROG event display allowed
NEvents_Frog  100
# Number of events to process
NEvents      -1          // -1 means 'all'

# input PDG tables
PdgTableFilename data/particle.tbl // table with particle pid,mass,charge,...

```

In general, energies, momenta and masses are expressed in GeV, GeV/ c , GeV/ c^2 respectively, and magnetic fields in T. Geometrical extension are often referred in terms of pseudorapidity η , as the detectors are supposed to be symmetric in ϕ . From version 1.8 onwards, the number of events to run will also be included in the detector card (NEvents).

2. The trigger card

This card contains the definitions of all trigger-bits. Cuts can be applied on the transverse momentum p_T of electrons, muons, jets, τ -jets, photons and the missing transverse energy. The following codes should be used so that DELPHES can correctly translate the input list of trigger-bits into selection algorithms:

<i>Trigger code</i>	<i>Corresponding object</i>
ELEC_PT	electron
IElec_PT	isolated electron
MUON_PT	muon
IMuon_PT	isolated muon
JET_PT	jet
TAU_PT	τ -jet
ETMIS_PT	missing transverse energy
GAMMA_PT	photon
Bjet_PT	b -jet

Each line in the trigger datacard is allocated to exactly one trigger-bit and starts with the name of the corresponding trigger. Logical combination of several conditions is also possible. If the trigger-bit requires the presence of multiple identical objects, the order of their p_T thresholds is very important: they must be defined in *decreasing* order. The transverse momentum p_T is expressed in GeV/ c . Finally, the different requirements on the objects must be separated by a `&&` flag. The default trigger card can be found in the data repository of DELPHES (data/TriggerCard.dat), as well as for both CMS and ATLAS experiments at the LHC. An example of trigger table consistent with the previous rules is given here:

```

SingleJet      >> JET_PT: '200'
DoubleElec    >> ELEC_PT: '20' && ELEC_PT: '10'
SingleElec and Single Muon >> ELEC_PT: '20' && MUON_PT: '15'

```

A.2.2. Running the code

First, create the detector and trigger cards (data/DetectorCard.dat and data/TriggerCard.dat). Then, create a text file containing the list of input files that will be used by DELPHES (with extension *.lhe, *.hepmc, *.root or *.hep). To run the code, type the following command (in one line)


```
me@mylaptop:~$ ./Delphes inputlist.list OutputRootFileName.root
                        data/DetectorCard.dat data/TriggerCard.dat
```

As a reminder, typing the `./Delphes` command simply displays the correct usage:

```
me@mylaptop:~$ ./Delphes
Usage: ./Delphes input_file output_file [detector_card] [trigger_card]
input_list - list of files in Ntpl, StdHep, HepMC or LHEF format,
output_file - output file.
detector_card - Card containing resolution variables for detector simulation (optional)
trigger_card - Card containing the trigger algorithms (optional)
```

A.3. Getting the DELPHES information

A.3.1. Contents of the DELPHES ROOT trees

The DELPHES output file (*.root) is subdivided into three *trees*, corresponding to generator-level data, analysis-object data and trigger output. These *trees* are structures that organise the output data into *branches* containing data (or *leaves*) related with each others, like the kinematics properties (E, p_x, η, \dots) of a given particle.

Here is the exhaustive list of *branches* availables in these *trees*, together with their corresponding physical objet and ExRootAnalysis C++ class name:

GEN TREE		
Particle	generator particles from HEPEVT	GenParticle
Trigger TREE		
TrigResult	Acceptance of different trigger-bits	TRootTrigger
Analysis TREE		
Tracks	Collection of tracks	TRootTracks
CaloTower	Calorimetric towers	TRootCalo
Electron	Collection of electrons	TRootElectron
Photon	Collection of photons	TRootPhoton
Muon	Collection of muons	TRootMuon
Jet	Collection of jets	TRootJet
TauJet	Collection of jets tagged as τ -jets	TRootTauJet
ETmis	Transverse missing energy information	TRootETmis
ZDChits	Hits in the Zero Degree Calorimeters	TRootZdcHits
RP220hits	Hits in the first proton taggers	TRootRomanPotHits
FP420hits	Hits in the next proton taggers	TRootRomanPotHits

The third column shows the names of the corresponding classes to be written in a root tree. The bin number in the unique leaf in the trigger tree (namely, `TrigResult.Accepted`) corresponds to the trigger number in the provided list. In addition, the result of the global trigger decision upon each event (i.e. the logical OR of all trigger conditions) is stored in the first bin (number 0) of this leaf. In Analysis tree, all classes except `TRootTracks`, `TRootCalo`, `TRootTrigger`, `TRootETmis` and `TRootRomanPotHits` inherit from the class `TRootParticle` which includes the following data members (stored as *leaves* in *branches* of the *trees*):

Most common leaves

```
float E;      // particle energy in GeV
float Px;     // particle momentum vector (x component) in GeV/c
float Py;     // particle momentum vector (y component) in GeV/c
float Pz;     // particle momentum vector (z component) in GeV/c
float PT;     // particle transverse momentum in GeV/c
float Eta;    // particle pseudorapidity
float Phi;    // particle azimuthal angle in rad
```

In addition to their kinematics, some additional properties are available for specific objects:

Leaves in the Particle branch (GEN tree)

```

int PID;           // particle HEP ID number
int Status;       // particle status
int M1;           // particle 1st mother
int M2;           // particle 2nd mother
int D1;           // particle 1st daughter
int D2;           // particle 2nd daughter
float Charge;     // electrical charge in units of e
float T;          // particle vertex position (t component, in mm/c)
float X;          // particle vertex position (x component, in mm)
float Y;          // particle vertex position (y component, in mm)
float Z;          // particle vertex position (z component, in mm)
float M;          // particle mass in GeV/c2

```

Additional leaves in Electron and Muon branches (Analysis tree)

```

int Charge        // particle Charge
bool IsolFlag     // stores the result of the tracking isolation test
float IsolPt      // sum of all track pt in isolation cone (GeV/c)
float EtaCalo     // particle pseudorapidity when entering the calo
float PhiCalo     // particle azimuthal angle in rad when entering the calo
float EHoverEE    // hadronic energy over electromagnetic energy
float EtRatio     // calo Et in NxN-tower grid around the muon over the muon Et

```

Additional leaf in the Jet branch (Analysis tree)

```

bool Btag         // stores the result of the b-tagging
int NTracks       // number of tracks associated to the jet
float EHoverEE    // hadronic energy over electromagnetic energy

```

Leaves in the Tracks branch (Analysis tree)

```

float Eta         // pseudorapidity at the beginning of the track
float Phi         // azimuthal angle at the beginning of the track
float EtaOuter    // pseudorapidity at the end of the track
float PhiOuter    // azimuthal angle at the end of the track
float PT          // track transverse momentum in GeV/c
float E           // track energy in GeV
float Px          // track momentum vector (x component) in GeV/c
float Py          // track momentum vector (y component) in GeV/c
float Pz          // track momentum vector (z component) in GeV/c
float Charge      // track charge in units of e

```

Leaves in the CaloTower branch (Analysis tree)

```

float Eta        // pseudorapidity of the tower
float Phi        // azimuthal angle of the tower in rad
float E          // tower energy in GeV
float E_em       // electromagnetic component of the tower energy in GeV
float E_had      // hadronic component of the tower energy in GeV
float ET        // tower transverse energy in GeV

```

Leaves in the ETmis branch (Analysis tree)

```

float Phi        // azimuthal angle of the transverse missing energy in rad
float ET         // transverse missing energy in GeV
float Px         // x component of the transverse missing energy in GeV
float Py         // y component of the transverse missing energy in GeV

```

The hits in very forward detector (ZDC, RP220, FP420) have some common data. In particular, the side variable tells in which detector (left:-1 or right:+1 of the interaction point) the hit has been seen. Moreover, some generator level data is provided for information, as the correspondance with the contents of the GEN tree is not possible. These generator-level data correspond to the particle kinematics (energy, momentum, angle) and identification (pid).

Common leaves for ZDC, RP220, FP420

```
float T          // time of flight in s
float E          // measured/smearred energy in GeV
int side         // -1 or +1
```

Generator level data

```
int pid;        // particle ID
float genPx;    // particle momentum vector (x component) in GeV/c
float genPy;    // particle momentum vector (y component) in GeV/c
float genPz;    // particle momentum vector (z component) in GeV/c
float genPT;    // particle transverse momentum in GeV/c
float genEta;   // particle pseudorapidity
float genPhi;  // particle azimuthal angle in rad
```

Additional leaves in the ZDChits branch (Analysis tree)

```
int hadronic_hit // 0(is not hadronic) or 1(is hadronic)
```

Additional leaves in the RP220hits and FP420hits branches (Analysis tree)

```
float S // detector position from IP in m
float X // hit horizontal position in m
float Y // hit vertical position in m
float TX // hit horizontal angle in rad
float TY // hit vertical angle in rad
float q2 // reconstructed momentum transfer in GeV2
```

The hit position is computed from the center of the beam position, not from the edge of the detector.

A.4. Running an analysis on your DELPHES events

To analyse the ROOT ntuple produced by DELPHES, the simplest way is to use the `Analysis_Ex.cpp` code which is coming in the Examples repository of DELPHES. Note that all of this is optional and done to facilitate the analyses, as the output from DELPHES is viewable with the standard ROOT TBrowse and can be analysed using the MakeClass facility. As an example, here is a simple overview of a `myoutput.root` file created by DELPHES:

```
me@mylaptop:~$ root -l myoutput.root
root [0]
Attaching file myoutput.root as _file0...
root [1] .ls
TFile**          myoutput.root
TFile*           myoutput.root
KEY: TTree      GEN;1      Analysis tree
KEY: TTree      Analysis;1  Analysis tree
KEY: TTree      Trigger;1   Analysis tree
root [2] TBrowse t;
root [3] Analysis->GetEntries()
(const Long64_t)200
root [4] GEN->GetListOfBranches()->ls()
OBJ: TBranchElement Event      Event_ : 0 at: 0x9108f30
OBJ: TBranch      Event_size  Event_size/I : 0 at: 0x910cfd0
OBJ: TBranchElement Particle   Particle_ : 0 at: 0x910c6b0
OBJ: TBranch      Particle_size Particle_size/I : 0 at: 0x9111c58
root [5] Trigger->GetListOfLeaves()->ls()
OBJ: TLeafElement TrigResult_   TrigResult_ : 0 at: 0x90f90a0
OBJ: TLeafElement TrigResult.Accepted Accepted[TrigResult_] : 0 at: 0x90f9000
OBJ: TLeafI      TrigResult_size TrigResult_size : 0 at: 0x90fb860
```

The `.ls` command lists the current keys available and in particular the three *tree* names. `TBrowse t` launches a browser and the `GetEntries()` method outputs the number of data in the corresponding *tree*. The list of *branches* or *leaves* can be displayed with the `GetListOfBranches()` and `GetListOfLeaves()` methods, pointing to the `ls()` one. In particular, it is possible to show only parts of the output, using wildcard characters (`*`):

```

root [6] Analysis->GetListOfLeaves()->ls("*.E")
OBJ: TLeafElement      Jet.E           E[Jet_] : 0 at: 0xa08bc68
OBJ: TLeafElement      TauJet.E       E[TauJet_] : 0 at: 0xa148910
OBJ: TLeafElement      Electron.E     E[Electron_] : 0 at: 0xa1d8a50
OBJ: TLeafElement      Muon.E        E[Muon_] : 0 at: 0xa28ac80
OBJ: TLeafElement      Photon.E      E[Photon_] : 0 at: 0xa33cd88
OBJ: TLeafElement      Tracks.E      E[Tracks_] : 0 at: 0xa3ccd0
OBJ: TLeafElement      CaloTower.E   E[CaloTower_] : 0 at: 0xa4ba188
OBJ: TLeafElement      ZDChits.E     E[ZDChits_] : 0 at: 0xa54a3c8
OBJ: TLeafElement      RP220hits.E  E[RP220hits_] : 0 at: 0xa61e648
OBJ: TLeafElement      FP420hits.E  E[FP420hits_] : 0 at: 0xa6d0920

```

To draw a particular leaf, either double-click on the corresponding name in the TBrowser or use the Draw method of the corresponding *tree*.

```
root [7] Trigger->Draw("TrigResult.Accepted");
```

Mathematical operations on several *leaves* are possible within a given *tree*, following the C++ syntax:

```

root [8] Analysis->Draw("Muon.Px * Muon.Px");
root [9] Analysis->Draw("sqrt(pow(Muon.E,2) - pow(Muon.Pz,2) + pow(Muon.PT,2))");

```

Finally, to prepare an deeper analysis, the MakeClass method is useful. It creates two files (*.h and *.C) with automatically generated code that allows the access to all branches and leaves of the corresponding tree:

```

root [10] Trigger->MakeClass()
Info in <TTreePlayer::MakeClass>: Files: Trigger.h and
      Trigger.C generated from TTree: Trigger

```

For more information, refer to ROOT documentation. Moreover, an example of code (based on the output of

MakeClass

To run the Examples/Analysis_Ex.cpp code, the two following arguments are required: a text file containing the input DELPHES ROOT files to run, and the name of the output root file.

```
me@mylaptop:~$ ./Analysis_Ex input_file.list output_file.root
```

One can easily edit, modify and compile (

make

A.4.1. Adding the trigger information

The Examples/Trigger_Only.cpp code permits to run the trigger selection separately from the general detector simulation on output DELPHES root files. A DELPHES root file is mandatory as an input argument for the Trigger_Only routine. The new *tree* containing the trigger result data will be appended to this file. The trigger datacard is also necessary. To run the code:

```
me@mylaptop:~$ ./Trigger_Only input_file.root data/TriggerCard.dat
```

A.5. Running the FROG event display

- If the FLAG_frog was switched on in the smearing card, two files have been created during the running of DELPHES: DelphesToFrog.vis and DelphesToFrog.geom. They contain all the needed pieces of information to run FROG.
- To display the events and the geometry, you first need to compile FROG. Go to the Utilities/FROG and type make. This compilation is done once for all, with this geometry (i.e. as long as the *.vis and *.geom files do not change).
- Go back into the main directory and type

```
me@mylaptop:~$ ./Utilities/FROG/frog
```

A.6. LHCO file format

The `*lhco` file format is a text-ASCII data format briefly discussed here. An exhaustive description is provided on <http://v1.jthaler.net/olympicswiki>. This section is based on this webpage. Only final high-level objects are available in the `lhco` format, and their properties are arranged in columns. Each row corresponds to an object in the event and all events are written after each other. Comment-lines starts with a hash `#` symbol.

#	typ	eta	phi	pt	jmas	ntrk	btag	had/em	dum1	dum2
0		57	0							
1	0	1.392	-2.269	19.981	0.000	0.000	0.000	4.605	0.000	0.000
2	3	1.052	2.599	29.796	3.698	-1.000	0.000	0.320	0.000	0.000
3	4	1.542	-2.070	84.308	41.761	7.000	0.000	1.000	0.000	0.000
4	4	1.039	0.856	58.992	34.941	1.000	0.000	1.118	0.000	0.000
5	4	1.052	2.599	29.796	3.698	0.000	0.000	0.320	0.000	0.000
6	4	0.431	-2.190	22.631	3.861	0.000	0.000	1.000	0.000	0.000
7	6	0.000	0.845	62.574	0.000	0.000	0.000	0.000	0.000	0.000

Each row in an event starts with a unique number (i.e. in first column). Row 0 contains the event number (here: 57) and some trigger information (here: 0. This very particular trigger encoding is not implemented in DELPHES.). Subsequent rows list the reconstructed high-level objects. Each row is organised in columns, which details the object kinematics as well as more specific information, such as isolation criteria or b -tagging.

1st column (#). The first column is the line number in the event. Each event starts with a 0 and contains as many lines as needed to list all high-level objects.

2nd column (typ). The second column gives the object identification code, or *type*. The different object types are:

- 0 for a photon (γ)
- 1 for an electron (e^\pm)
- 2 for a muon (μ^\pm)
- 3 for a hadronically-decaying tau (τ -jet)
- 4 for a jet
- 6 for a missing transverse energy (E_T^{miss})

Object type 5 is not defined. An event always ends with the row corresponding to the missing transverse energy (type 6).

3rd (eta) and 4th (phi) columns. The third and fourth columns gives the object pseudorapidity η and azimuth ϕ . This latter quantity is expressed in radians, ranging from $-\pi$ to π .

5th (pt) and 6th (jmass) columns. The fifth column provides the object transverse momentum (p_T in GeV/ c) or energy (E_T in GeV), while the invariant mass (M in GeV/ c^2) is in the sixth column.

7th column (ntrk). The seventh column reports the total number of tracks associated to the objects. This is 0 for photons, ± 1 for charged leptons including taus (where the sign reports the lepton measured charge) and a positive number (≥ 0) for jets.

8th column (btag). The eighth column tells whether a jet is tagged as a b -jet (1) or not (0). This is always 0 for electrons, photons and missing transverse energy. For muons, the closest jet is searched for, in terms of ΔR . The integer-part of the quoted number is the row-number (column 1) of this jet.

9th column (had/em). For jets, electrons and photons, the ninth column is the ration between hadronic and electromagnetic energies in the calorimetric towers associated to the object. This is always 0 for missing transverse energy. For muons, this number (aaa.bb) reports two values related to the muon isolation (section 3.1). The integer part (aaa) is transverse momentum sum P_T (in GeV/ c) and the fractional part (bb) is the energy ratio ρ_μ .

10th and 11th columns (dum1 and dum2). The last two columns are currently not used.

Warning. Inherently to the data format itself, the `*lhco` output contains only a fraction of the available data. Moreover, dealing with text file may have various drawbacks, such as the output file size and the time needed for its creation. Whenever possible, working on the `*root` output file should be preferred.